パーソナル・コンピュータを用いた テキスト・データベースの作製

原　田　龍　二

0．言語の研究では，実際のテキストから，ある構文や単語の用例を採取したり，それらの頻度を調査したりすることがよくある。又，文学研究でも，ある作家の作品のコンコーダンスを作りたいと言う声をよく耳にする。本論では，そうした需要に応えるテキストデータベースをパーソナルコンピュータ上で作製するときの作業手順と設計上の留意点を述べることにする。

テキスト分析の最も基本となるのは，テキスト中の全ての単語を切り出し，それらの一つ一つに参照用の番号（インデックス番号）を付け，それをアルファベット順に並べる作業である。こうして，全ての単語とそれらの出現頻度の辞書を作る。これを用いると，インデックス番号を手掛りに原文を参照し，その語を含む文とその前後の文を，一定範囲で出力することも容易にできる。著者は，この作業を，タガログ語の小説（約300ページ6万語）について行ない，構文の研究や，読解時の用例の参照の手助けに使用している。辞書が英語のようにたくさんないタガログ語では重宝している。

作業手順は次のようになる。以下にそれぞれの詳細を述べる。

（1）　①入力
　　②エディット
　　③ファイル変換
　　④単語切り出しとインデックス番号付け
　　⑤ソート

1. 入　力

全作業の中で，最も労力を要し，かつ大切なものである。逆にこれさえやってしまえば，後の処理はどうにでもなると言える。

つい最近までは，これだけは，手でやらなくてはならないと決まっていたが，今日では，イメージスキャナで紙面を読み取り，文字を一つ一つ判読する OCR (Optical Character Recognition（光学的文字認識）) と呼ばれる技術が使用できるパーソナルコンピュータが登場し，1ページが2・3分で文字コードになってしまう。著者も上述のタガログ語小説の入力には，三分の一はこれを用いた。

OCR の詳細は他へ譲ることとし，ここでは，手入力について述べる。

1.1.　ワープロ又はエディタの選択

入力はワープロ又はエディタを用いて行なう。ワープロもエディタも，コンピュータに文字を入力し編集する道具であることに変わりはないが，エディタは，あくまで，コンピュータで利用する文書（プログラムなど）を画面上で作ることが主眼であるのに対して，ワープロの方は，入力した

文書を紙に印刷して，きちんとした紙上の文書を作ることが目的の道具である。エディタによって作られるファイルは，そのままでデータファイルとして利用できる ASCII 形式の汎用ファイルであるが，ワープロで作られるファイルは，印刷関係の制御コードや，左右揃えをするための特別なスペースが入っているのが普通で，そのワープロ専用のものであり，データファイルとしてはそのままでは利用できない。これを利用するためには，一度ファイル変換を行ない，ASCII ファイルを作らなくてはならない。

それでは，エディタを用いた方が良いかと言うと，そうは簡単に決まらない。次のことがらを考慮して決定すべきである。

①入力がしやすく，速くできること。当たり前のことだが，例えば，長い英文をパラグラフ単位で入力するときには，自動的に改行して，行からあふれ出た単語を次行へ送ってくれる（ワードラップする）方が，一行一行改行して打って行くより速い入力ができる。この機能があるのはワープロである。

②スペルチェックができること。タイプミスが発見され，正確度が高まる。入力で最も大切なのは，正確に入力することである。一字でも入力ミスがあり，それを訂正すれば，単語のアルファベット順と頻度が変わるので，入力以降の全ての作業（単語切り出しとソート）を始めからやり直さなくてはならない。正確な入力をし，この作業のやり直しを減らしたいものである。最近の英文ワープロには，スペルチェカが付いているので，これを利用したい。著者はタガログ語の入力をしたので，英文スペルチェカの恩恵には浴さなかった。

③印刷がしやすく、結果が見やすいこと。これにはワープロに軍配が上がる。エディタには、専用の印刷機能を持たないものすらある。本来，テキスト処理では，入力データを印刷することが主目的ではないが，ばらばらになった単語に付けられたインデックス番号を見て，原文の文脈を参照することもあるので，印刷物も必要である。また，次節で述べる編集作業は，画面上でもできないことはないが，作業能率の上からも，目の健康のためにも，初めは紙の上でやるのが良い。

④扱えるデータ量（行数）が多く，動きが速いこと。本来この点で優れているのはエディタである。しかし大量のデータは分割して入力するのが普通なので，ワープロでも問題は起こらない。著者は，6万語（6万行）の単語レコード全てを一つのファイルで見たいときにエディタを使うことがある。

⑤ファイルの可搬性，融通性があること。エディタは ASCII ファイルを作るので，この点はエディタの方が優れている。しかしワープロにも，ワープロ専用ファイルから ASCII ファイルを作る機能を持ったものも多いので，これを利用して ASCII ファイルを得ることができる。また，ワープロ専用機（主に日本語ワープロ）にも，MS-DOS ファイルへのファイル変換機能を持ったものも多いので，入力だけをワープロ専用機で行なうこともできる。変換された MS-DOS ファイルは ASCII ファイルである。[1]

以上のことを勘案し，始めて入力を考える場合はワードスターに代表される，英文ワープロを使うと良いであろう。ワードスターはプリント機能の一部として ASCII ファイルへの変換機能を持っているので，本稿ではそれを使うことにする。

1.2. 入力時のファイル分割

長大なテキストを入力するときに，その全てを一つのファイルに入れるようなことをしてはならない。ワープロによっては，一度にあまり大量のデータを扱えないだろうし，仮にできたとしても，テキスト内でのカーソルの移動に時間がかかる上に，自分がテキストのどの辺にいるのかわか

らなくなってしまうことすらある。また，ファイルの保全上も好ましくない。一たび事故（停電，メモリーエラー，ハードディスクのクラッシュ，フロッピーディスクの汚損等）があれば，長大ファイルの全てが，一瞬にして失われてしまうこともあり得る。

通常は以上のことを考え，長大なテキストは，複数のファイルに分割して入力し，編集などもファイルごとに行ない，本当に必要なときのみに，それらを一本のファイルにまとめる。

著者はタガログ語小説300ページの入力にあたっては，全30章（各章約10ページ）を章ごとに入力し，そのそれぞれについて編集し，ファイル変換，単語切り出し，ソートまでを行ない，最後に，ソートされた30ファイルを融合（マージ）して，全体で一つの大きな辞書を作るようにしている。こうしておけば，ある章に誤りが発見され，それを修正したときは，その章のみについてソートまでを行ない，最後に，既にソートしてある他のファイルとのマージを行なうだけで良い。

どの程度のファイルの大きさが適当かは，パソコンとワープロの能力や，扱う人の経験によって違ってくる。一つの目安としては，15ページぐらいになったら別のファイルにすると良いのではないかと思う。著者の場合は，どのファイルも8ページ前後であった。もう一つの目安としては，ワープロ使用中に，カーソルを一番速い方法で，入力テキストの先頭から，一番最後に一気にジャンプさせたときに（ワードスターでは^QC など），待ち時間が長いと感じられたら，それは大きすぎるファイルである。また，ワードスターでは大きなファイルを扱うと，Large File と言うメッセージが画面に常に表示されるようになる。これも参考になろう。

2. エディット

入力後のテキストは，体裁を整え後の処理をしやすくするために，次のことを行なう。これらはワープロのエディット（編集）機能を用いて行なう。

（2） ①ミスタイプの修正
②文字セットの確認
③特殊文字処理
④ハイフン処理
⑤ページマークとパラグラフマークの書き入れ
⑥テキストインフォメーションの書き入れ（コメント行）

2.1. ミスタイプの修正

英語の場合，スペルチェッカが利用できるので比較的楽に行なうことができる。スペルチェッカが利用できない場合は，簡単な自作のスペルチェッカを作ることもできる。これは辞書を使うものではなく，予想される明らかなミスを見つけるものである。例えば，母音字（a,e,i,o,u）の全く無い単語を見つける（無論，CNN のような語は，ミスとされてしまう），5個以上の子音字連鎖はまれである（英語，タガログ語），語頭の子音字連鎖は4つ以上はあり得ない（英語）等を調べるだけでも，ある程度の誤りを見つけることができる。

2.2. 文字セットの確認

文字セットとは，当該テキストで使用される，又はされない文字や記号のそれぞれの集合である。これが何であるかを確認しておくと，後の作業に便利である。

以下の三つの区別があれば良い。ASCII コード（16進数）20H から 7FH までの文字を全てそれ

らのいずれかに一義的に入れておく。

（3） ①使用文字　単語文字セット　{ A..Z　a..z　－　′ }
②使用文字　非単語文字セット　{ . , “ ; : ? ! ( ) [ ] スペース }
③不使用文字セット　{ ¥ # $ % & = ^ | @~ } { * + < > _ }

単語文字セットの文字は単語そのものになる文字で，単語の切り出しをするときに参照され，これ以外の文字は捨てる。アルファベット大文字・小文字は全て入れておく。その他，英語，タガログ語では，ハイフン，アポストロフィ（シングルクォート)が入る。これは下の例のような単語の一部になっている。

（4） 英語：　can′t　well-known
タガログ語：　′ko(=ako)　mag-alis

ここで注意すべきは，アルファベット大文字・小文字については，実際に使われていない文字が含まれていても何らさしつかえないが，逆に，使用されているのに入っていないと不都合がおきる。従って，アルファベットは全て入れておくのが安全である。一方，単語文字としての記号は，厳密に選ばれなくてはならない。もしハイフンが単語文字セットに含まれていないとすると，(4)の例の well-known，mag-alis は単語切り出しのときに，ハイフンの所で切れていると判断され，前半部と後半部が，別々に辞書に登録されてしまう。

②の非単語文字セットは，テキスト中に現われるが，単語の一部ではない文字である。通常，{. , “ ？！スペース} が必ず入るであろう。②に入らない残りの記号類が③の不使用文字セットとして残る。これらはテキスト中に全く現われない。不使用文字は，インデックス番号付けに使用する，ページやパラグラフのマークや，テキスト情報書き込み用のマークとして確保しておく。

不使用文字セットには，使われている文字は絶対に入っていてはいけない。したがって，疑わしきは使用文字の非単語文字セットに入れておくのが良い。こうした区別を厳密にやるには，プログラムを組んで，それぞれの記号や文字について分布を調べれば良いが，簡単に調べるには，ワープロの検索機能を用いて，当該文字の有無を調べれば良い。又，経験的に予想できる範囲で不使用文字セットを決め，残りを非単語文字セットに入れておくのでも十分である。例えば，韻文には記号類は使われないだろうし，普通の小説でも，@く＞｜％等も使われないはずである。いずれにしても４・５個ぐらいを選んで確保しておくと良いであろう。

2.3. 特殊文字，特殊単語の処理

原文テキスト中にある，特殊な記号や，テキスト言語とは異なる外国語の単語に，マークを付ける。

例えば，ギリシャ文字α，β，γに対して，@a，@b，@g　のように代わりの文字（文字の組み合わせ）を決めておく。これらは，当然，実際に使われている単語や，他のマークと重複してはならない。前節で確定した，不使用文字セットから文字を選び，それと他の文字を組み合わせると，重複は起こらない。

外国語の単語に関しても，一定の処理をしておくと良い，著者が使用した，タガログ語小説の原文では，純粋な英語及び英文は，イタリック表記になっているので，それらの英単語の先頭にマー

クをつけ，単語切り出しのときにタガログ語単語と区別するようにしている。

もし必要ならば，固有名詞に関しても同様の処理をしても良い。但し，固有名詞を見分けるのはこの段階では手作業でやらなくてはならない。

2.4. ハイフン処理

単語の切り出しや，単語検索で，意外にやっかいなのがハイフンの扱いである。英語でもタガログ語でも，もともと単語の一部としてあるハイフンと，行末で単語を切るために入れたハイフンとがある。さらに後者は，原文にあるものと，入力時に入るものとがある。まとめると次のようになる。

（5） ①始めから単語の一部としてあるハイフン　例 well-known, mag-alis
　　a．原文で行中（行末でないところ）にありつながっている
　　b．原文で行末にあり単語を分けている
　　②単語（スペリング）分割のためのハイフン
　　a．原文に始めからあるもの（原文では必ず行末にあり単語を分けている）
　　b．入力時にワープロが自動的に入れたもの

これらのハイフンを，このまま後の段階まで残しておくと，単語切り出しや検索が正しくできなくなり，又，できたとしてもプログラムが複雑になることが予想されるので，必要最低限のもの以外は，取ることにする。

ハイフン処理に限らず，テキスト処理では，記号や文字の用法は，一義に決まっていることが原則である。その意味からも，②a，②b，が存在することは好ましくない。なぜなら，①a，①bのハイフンは，前述の単語文字セットに入るが，②a，②bのハイフンは，非単語文字セットに入るので，同一文字（ハイフン）が，二つのセットに入っていることになるからである。したがって，ハイフン処理では，①a，①bのみを残し，それ以外は取り除くことが必要である。

②bに関しては，ワープロの内部で，特殊なコードが使用されているので，ファイル変換時に取り除くこともできるし，エディット時に検索・置換機能を用いて，手作業で取り除いても良い。

エディット時の主たる作業は，(5)の②aと①a，①bを区別して，②aを取り除くことである。(5)のそれぞれのハイフンの原文及び入力ファイル上での所在は次のように対応している。

（6）

	原文	入力ファイル上（ワープロ上）
①a．	行中	行中又は行末
①b．	行末	行中又は行末
②a．	行末	行中又は行末
②b．	無し	行中又は行末

(6)からわかるように，原文では行末にあるものが，ワープロのファイル上では必ずしも行末に来ず，行中に来てしまうことがあるので，①bと②aの区別をする時は，原文の行末を見てチェックすれば比較的わかりやすい。

ワープロファイル上でハイフンが必ずしも行末に来ないのは次の理由による。

（7）　a．原文の一行の文字数と，ファイル上の一行の文字数が一致していない
（一致しないのが普通）
b．一行の文字数が変更され，行末に来る単語にずれが生じた

以上のようにハイフン処理は複雑な要素を含んでいるが，入力時に次のことを守れば，エディット時のチェックは簡単ですむ。

（8）　a．事前に原文の①a，①bのハイフンのみに印を付け，入力時には，その印のハイフン以外は決してハイフンキーを打たない[2]
b．ワープロの自動ハイフン入力はoff にする（②bがはいらなくなる）
c．行揃え（パラグラフリフォーム）を行なっても①aのハイフンが①bにならないようにする（ワードスターではその語の先頭で^OEを打ち語が分割されないようにする）

こうして入力すれば，(7a)で印を付けたハイフンがきちんと入力されているかさえチェックすれば良いことになる。さらに（8c）の処理を行なっておけば「いかなる単語も二行にまたがって存在しない」ものとして後の単語切り出し作業を行なうことができる。

2.5.　インデックスマーク付け

著者は，前述のタガログ語小説のインデックス番号として，原文のページ番号と，そのページでのパラグラフ番号を，切り出された単語に付している。これらの番号は，入力されたテキストから，プログラムで単語をばらばらに切り出す際に，プログラムが自動的に数えて行く。従って，番号自体をテキストに直接入力する必要はないが，ページやパラグラフの切れ目に，プログラムにそれを知らせるマークを入れておく必要がある。

（9）　例1　　ページの切れ目に　＃　を入れる
パラグラフの先頭に　¥　を入れる

例1のように一文字のみを用いて，ページやパラグラフのマークとする場合は，それらのマークは，不使用文字セットから選ばなくてはならない。

（10）　例2　　ページの切れ目に　〈＃〉　を入れる
パラグラフの切れ目にスペースを5個入れ，これをマークとする

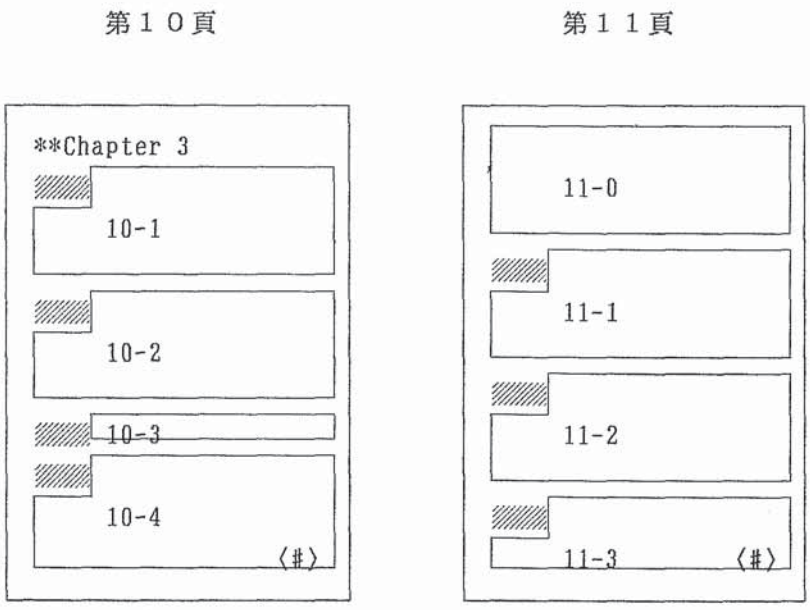
例2のように，2文字以上を組み合わせてマークとする場合は，単語文字セット以外の文字が使用できる。著者は，原文で段下げが行なわれている部分は，会話文の文頭（どんな短い文でも）を含め，1パラグラフとしているので，この段下げ部のスペース5個をパラグラフマークとしている。この方が，目ざわりなマークを入れるより，テキストのプリントアウトが見やすくなる。スペースは使用文字の非単語文字セットの記号である。〈＃〉は，いずれも不使用文字セットの記号であるので，〈，＃，〉のいずれか一つでもマークとしては用が足りるが，やはり見やすさの点から3文字を使用している。どのような組み合わせでも良いが，いずれも，単語切り出しのプログラムでは，これらをページやパラグラフのマークとして捜すことになる。但し，(9)の例の方が，一文字

のみをチェックすれば良いので，単語切り出しのプログラムは単純になる。(付録1のプログラムはどちらにも対応できる)

2.5.1. 実際のインデックスマーク付け

(11)は，(10)の方法でのインデックスマーク付けの例である。▒で示した段下げ部分（スペース5個）がパラグラフマークである。章のタイトルや Chapter 3 のような見出しは，通常単語データとはせず，1パラグラフと数えないので，2．6節で述べるコメントマークを付け，単語切り出し時には無視されるようにする。[3] 10-1～11-3はインデックス番号を示す。それぞれ，第10ページ第1パラグラフ～第11ページ第3パラグラフの意味である。11ページ始めの11-0は，第11ページ第0パラグラフ，即ち，前ページからの続きのパラグラフを意味する。(下図は原本のイメージである。原本のページの切れ目と，ワープロが管理する文書としてのページの切れ目は一致しないのが普通である。したがってページマーク（〈#〉）はワープロのページの切れ目に来るわけではない。)

(11)

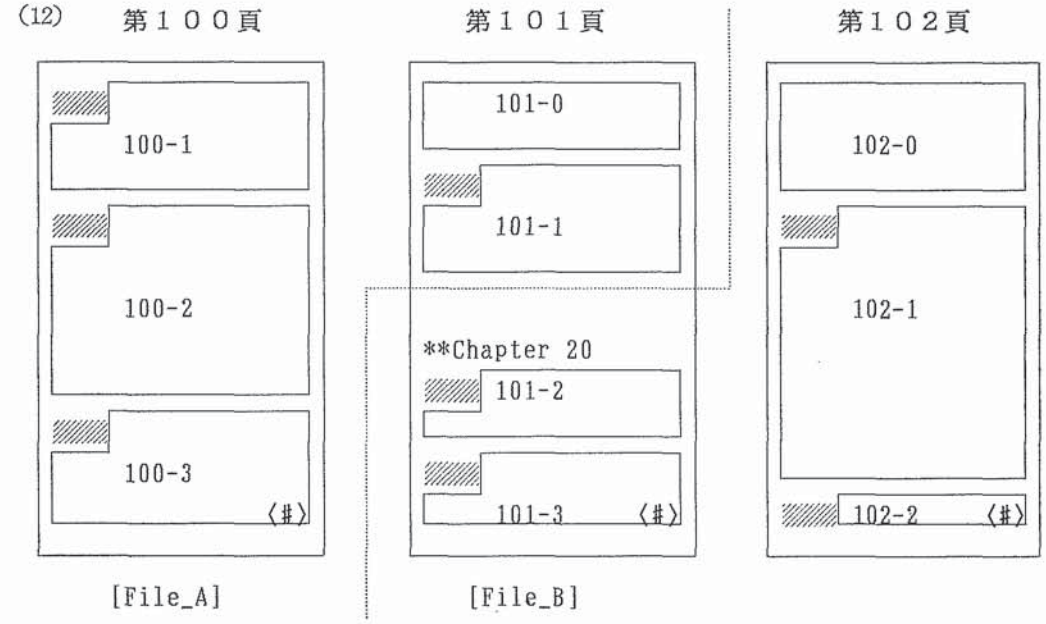


ページマークを入れる場所は，ページの終わりでも始めでも良い。上例のようにページの終わりに入れた場合は，インデックス番号付けのプログラム（単語切り出しプログラムの一部）のページカウンタの初期値は，数え始めるページ番号と同じになるが，ページの先頭にページマークがある場合は，それより1減じたものとなる。ページの終わりにマークを入れる方法で，ファイルの終わりがページの切れ目であるときは，最後にページマークを入れるのを忘れてはならない。

ページマークは，プログラムのページカウンタの値を1マークごとに1加算させるためのものであるので，原文にテキスト文字のない図や写真などのページがあるときは，ページマークだけを入れておかなければならない。

入力時にファイル分割を行なって，ページ中程でファイルが切れるときも，マークの入れ方に注意が必要である。

例⑿では，101−1パラグラフの終わりが［File-A］の終わりであるが，ここにはページマークは入れない。［File-B］は101−2から始まるが，第101ページのページマークは101−3パラグラフの末尾に入れる。



2.5.2　インデックス番号の初期値について

このことからわかるように，ファイル分割をしたばあいは（長大なテキストは必ず分割する(1.2参照)），ページカウントはもとより，パラグラフカウントも，常に1から始まるわけではないので，インデックス番号の数え始めの値は，ファイルごとに与えてやらなくてはならない。⑿の［File-B］の場合は，プログラムに101ページの第2パラグラフが始めのインデックス番号であることを知らせなくてはならない。

2.5.3.　ページにまたがる単語

この段階では，ハイフン処理がされているはずである。原文で，ページの切れ目に，ハイフンを単語の一部として持つ語（⑹①ａ，①ｂ）が来る場合は，(13)の例のように，ページマークはその語全体の後に入力する。こうしておくと，well-known には95−2のインデックス番号が付く。

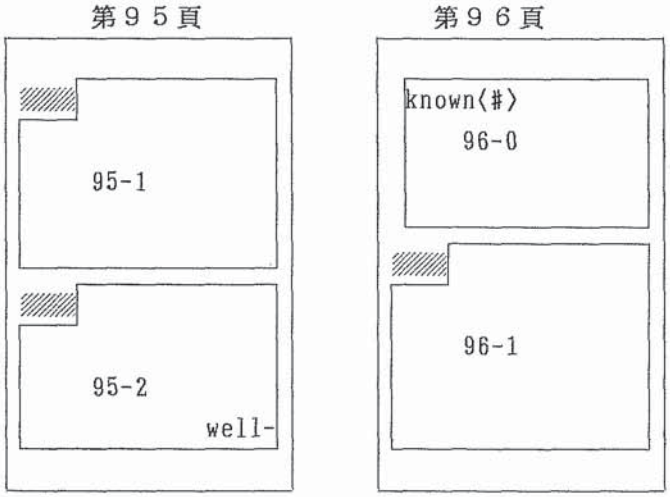
2.5.4.　マークの検査

最後に，意図した個数どおりのマークが入っているかを検査する。これを怠ると，インデックス番号の誤りが発見されたときに，原因を突き止めるのに苦労する。プログラムの誤りだと思い，一生懸命プログラムの手順を考え直していると，案外テキストファイルのマークの仕方や個数が間違っていることがある。

この検査には，ワープロの検索機能を用いると見落としがなくなる。目で見ただけでは，スペース5個と6個は見分けにくいが，ワープロは誤らずに見分けてくれる。ワードスターでは，同じ文

字を一つずつ順に連続的に検索できるので，著者はこれを使い，キーを打つごとに「一つ，二つ」と数えて検査をしている。実際に数えた値は，どこかにメモするか，ファイル内にコメントとして記録しておくことも忘れてはならない。

(13)



2.5.5. インデックス番号の単位について

上例では，パラグラフが単位になっていたが，行を単位とすることもできる。この場合，200ページ15行目，のようにページ数と併用する方法，650行目，のように全体を通して番号を付ける方法などが考えられる。

行を単位とする方法の利点は，行はプログラムでの読み取りの単位となっているので，特別なマーク付けが不要であることである。特に，パラグラフがはっきりせず，大きな区切りを区別しにくいときに有効である。しかしワープロで入力した行数と，原文の行数は一致しないのが普通なので，ワープロ上の行数を見て原文の特定箇所を参照するのは困難である。それゆえ，参照用に，行数の通し番号入りのテキストをプリントアウトすることが必要である。しかしこれは，テキストの誤入力訂正があり，行の組み替えが少しでもあったときは，再プリントしなくてはならないので，あまり良い方法とはいえない。

詩や韻文では，行の区切りがはっきりしているので，テキスト入力も行単位で行なうであろうから，原文と入力テキストとの行のずれはないはずである。このようなテキストでは，行単位のインデックス番号付けが有効である。

行単位の場合も，ハイフンを含む語が，２行にわたらないようにしておくと，後の処理がやさしくなる。

2.6. テキストインフォメーションの書き入れ（コメント行）

ここまで入力テキストが整ったところで，そのテキストに関するインフォメーションを，コメント行としてファイルの先頭に入れる。一例を(14)に示す。

①はこのファイルが何ページの何段落目から始まっているかを記したものである。インデックス号付けのプログラムにはこれと同じ値をカウントの開始値として与える。CDR はタイトルの略号である。②は章のタイトル，③〜⑦は出典に関する情報である。将来，テキストの種類が増えたときに備えて入れておく。

⑧はファイルの修正がいつ行なわれたかの記録である。誤入力が発見された時に，その日付けを

メモしておき，⑧の日付けと照合すれば，その誤りが，既にファイル上で訂正されたかが一目でわかり便利である。その他に，⑨のようにメモやコメントを必要に応じて入れることができる。ここでは，2．5．4節でも確認されたページとパラグラフのマークの数をメモとして書いている。Chapter 3 のような行にも**を付けておく。

(14) ① **CDR 101.2-115.3
② **9. Inisasyon
③ **Title: Canal de la Reina
④ **Author: ARCEO, Liwayway A.
⑤ **Publisher: Ateneo de Manila University
⑥ **Place: Quezon City
⑦ **Year: 1985
⑧ **Last correction:911130
⑨ **14 Page marks 58 paragraph marks

これらの記載事項は，単語切り出しのときには，テキストの本文の単語とは区別され，出てこないようにプログラムしなくてはならない。上例の**は，そのためのマークである。「**で始まる行は，コメント行であり，以下の文字を無視する」とプログラムすれば良い。このマークは，＊ひとつだけでも，¥のような記号でもよいが，文字セットを参照しながら，他のマークと重複しないように，決めなくてはならない。

3. ファイル変換

ワープロで作った文書ファイルは，通常そのワープロ専用で，テキストデータファイルとしては利用できないので，これを汎用の ASCII 形式のファイルに変換する必要がある。

ASCII 形式のテキストファイルは，内部コードとして，16進数 20 H から 7FH までの文字コードと，0DH(復帰)，0AH(改行) の制御コードを用いたファイルである。先に定義した文字セットの要素は，全てこの範囲に納まる。

ワードスターを使用している場合は，プリント機能より，プリンタとして ASCII を選ぶことにより，あたかも，紙上に文字を出力するかのごとく，ディスク上のファイルに，ASCII コードを出力してくれる。標準ファイル名は ASCII.WS となるが，複数のファイルを次々と変換するときは，別の名前をそれぞれのファイルに付けなくてはならない。

ファイル変換時には，ワードスターの変換前のファイルの始めに（テキストインフォメーションより前に），ドットコマンドで次の指定をする。

(15) .MT 0
.MB 0
.PO 0

それぞれ順に，上下，左部の余白をゼロに取るということで，これにより，不要な余白の挿入がなくなる。尚，出力ファイル（ASCII ファイル）には，改ページのコードは入らない。

その他に，出力ファイルでは，次のものが取り除かれたり，変換されたりする。

(16)

ドットコマンド	→無くなる	ソフトハイフン（行中）	→無くなる
コントロールコード	→無くなる	ソフトハイフン（行末）	→ハードハイフン
ソフトペース	→ハードスペース	ハードハイフン	→ハードハイフン
ソフトリターン	→ハードリターン	ページ番号	→入らない

本稿の手順に従えば，今までの作業で，ソフトハイフンは全て取り除かれており，また，行末にハイフンは来ないようになっているので，ファイル変換で，ハイフンの様態は変化しない。

出来上がったファイルは，ワードスター上で，非文書（Non-Document）ファイルとして開き，内容を見ることができる。念のため，上，下，左の余白がないこと（特に，左余白のスペースが入っていると，パラグラフカウントが正しく行なわれない），行末にハイフンが来ないことを確認すると良い。

MS-DOS のコマンドレベルから，type コマンドを使って変換前と変換後のファイルを画面に表示させれば，両者の違いがわかるであろう。（例，A>type［ファイル名］）

4. レコードのデザインと単語の切り出し

いよいよ，できあがった ASCII ファイルを，プログラムで読み出して，単語を取り出すことになるが，その前に，個々の単語を単位に，どのような情報をどのようにまとめて，一つのレコードにするかを決めておく必要がある。

4.1. レコードの内容

レコードは，データベースの単位となるもので，本稿で作る１レコードの基本的な構成は，下例のように，データ（単語）フィールドと，２個のインデックスファイルからなる。（レコード内の一つ一つの区切りを，フィールドと言う。）

(17)

```
mo'y, 23, 3
kanila,128, 5
biru-biro,133, 7
```

biru-biro：データフィールド
133：インデックスフィールド１
7：インデックスフィールド２

最終的な，レコードの形式は，後に使用するデータベース管理プログラムによって異なるが，始めにテキストから単語を切り出して作製するレコードは，ASCII 標準形式と呼ばれる，可変長のレコードにしておくのが良い。

この形式のレコードは，フィールドの区切りとしてカンマ(,)を使用し，各レコードの終わりには改行復帰記号（[CR]+[LF]（0DH+0AH））が入っている。数字は，２進数バイト表現ではなく，ASCII 文字の数字（30H～39H）を用いて，10進数の文字列表現として表わされる。

ASCII 形式のレコードファイルは，ASCII 形式のテキストファイルの一種であるので，ワープロやエディタを用いて直接見たり書いたりすることができる。従って，実際のファイルでも，(15)の例と全く同じイメージで，テキストと同じように一行一行のレコードが書かれている。

4.2. フイールドの区切りについて

ASCII 形式のレコードファイルでは，カンマが区切り文字として使用されるので，単語（文字）データそのものに，カンマが含まれるフィールドがある場合は注意が必要である。通常このような場合は，そのフィルード全体をダブルクォーテーション（“）で囲んでおく必要がある。こうした区切り記号の使い分けを誤ると，出来上がった ASCII 形式のレコードファイルを，データベース管理プログラム（dBASE,PARADOX等）に引き渡すときに，正しくデータが渡されなくなる。自分が使用するデータベース管理プログラムが，ASCII ファイルの区切り記号として何を使用しているかを良く知って，あらかじめ対策を取っておく必要がある。

著者が使用した，PARADOX というデータベース管理プログラムでは，カンマ(,)，ダブルクォーテーション（“），シングルクォーテーション（'）の三者が，区切り記号として使われるため，アポストロフィー（ASCII 文字ではシングルクォーテーションと同じ文字を使う）を単語の一部に使う，英語やタガログ語のデータに不都合が生じたが，単語フィールド全体をダブルクォーテーションで囲むことで解決した。(例“mo'y”,23, 3）ちなみに，ダブルクォーテーション自身が文字データに含まれている時は，ダブルクォーテーションを二つ重ねるのが通常の対処法である。これらの作業は，既に確認されている文字セットの内容を参照しながら行なう。

どうしても区切り文字と，単語文字の調整がつかないときは，一時的にその単語文字を他の文字に置き換え，後でまたもとの文字に置き換えるという手法を用いる。(例，'を%に置き換える）置き換える文字は，不使用文字セットより選べば良いが，置き換えと復元が一義的に行なわれるよう注意しなければならない。置き換え作業は，単語切り出しのプログラム中で行なっても良いが，レコードファイルは ASCII ファイルで作ってあるので，エディタやワープロを使って，直接レコードファイルに対して作業を行なうこともできる。

4.3. 単語切り出しの実際

これまでの作業で，テキストファイルは，かなり形が制約され，整ったものになっているので，単語切り出しの手順は，下に示すように簡単なものになる。(PASCAL で書かれた実際のプログラムを付録１に載せる。）以下に(18)のフローチャートに沿って手順の概略を述べる。

①[初期設定，ファイル操作] 変数，及び単語文字，単語終わり検出用のフラグ等の初期値を設定する。入出力ファイルを開く。

②[インデックス番号開始値入力] 何ページの何パラグラフからインデックス番号を付けるかをここで与える。

③[一文字読み出し] テキストファイルから一文字読み出す。それが単語文字か，マークの一部か等を，以下に続く手続き（procedure）で毎回チェックし，それぞれ見合った処理をする。

④[コメント行処理] コメント行（本稿では**で始まる行）があったとき，それに対する処理をする。ここではコメント内容を画面に表示するのみで，後は捨てられる。無論，単語データとはならない。

⑤[パラグラフチェック] パラグラフマークが見つかった時はパラグラフカウンタを１増ずる。

⑥[単語文字チェック] 処理中の一文字が単語文字セット（プログラム中では集合型 WordChar として定義）に含まれるかをチェックし，含まれるときは単語データとして，ファイルに書き込む。単語文字はこのようにして一文字ずつ書かれ，しだいに一単語になる。

⑦[単語終わりチェック] １単語が終ったことを検知し，その後にインデックス（ページ，パラグ

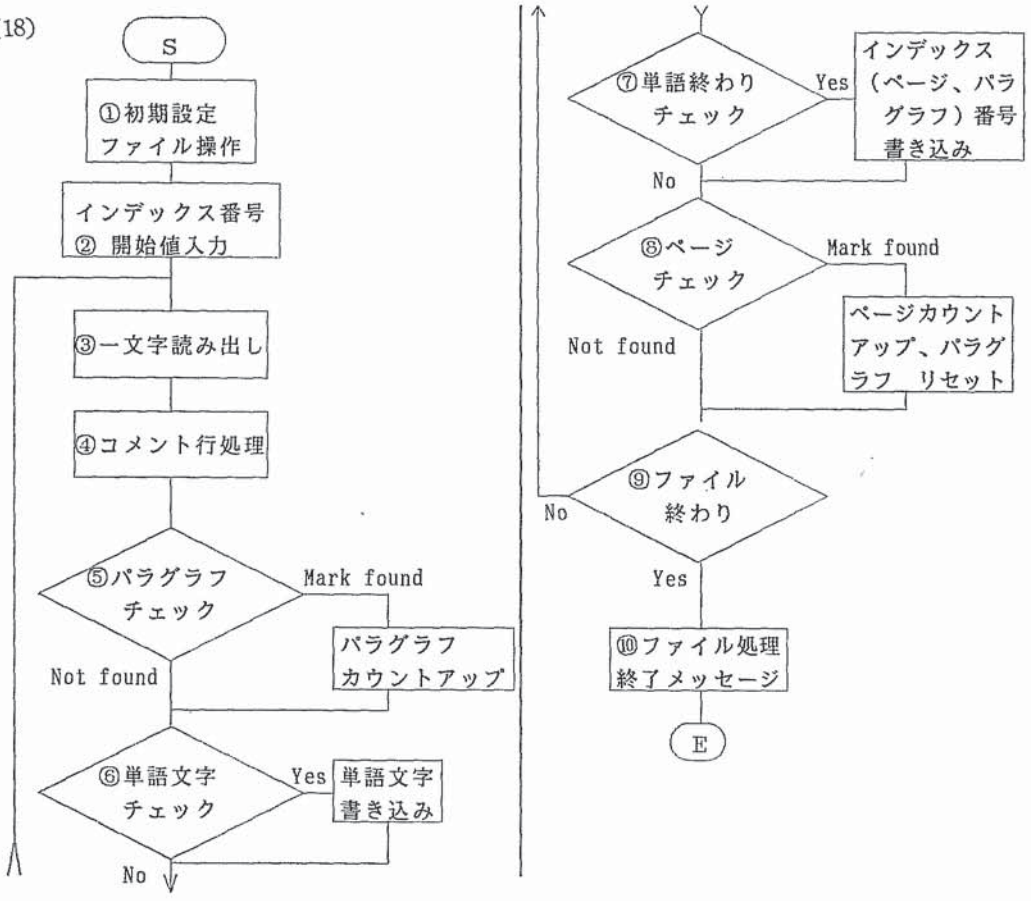
ラフ）番号をレコードの設計通りの順にカンマで区切って書き込み，１レコードを終える。

⑧[ページチェック] ページマークが見つかったときは，ページカウンタを１増ずる。パラグラフカウンタはリセット（ゼロに）する。ページマークをページの先頭に入れる場合は，パラグラフチェック⑤の前におく必要がある。２．５．１節。参照

⑨[ファイルの終わり] ファイルが終わるまで一文字ずつ全ての文字について以上の③～⑧までを繰り返し，ここでファイルの終わりを検知したなら，⑩入出力ファイルを閉じ，終了メッセージを出してプログラムを終える。

付録２にサンプルの入力テキストファイル（変換済み ASCII ファイル），付録３に単語切り出しとインデックス番号付けの結果を示す。

(18)



5. ソート

得られた単語レコードは，テキスト中の出現順になっているので，これを単語のアルファベッド，ページ番号，パラグラフ番号の順にソートして辞書の体裁にする。

一口にソートといっても様々な方法が考えられるが，とりあえず一番簡単なのは，データベース管理プログラムで行なうことである。データベース管理プログラムは，こうした仕事の専門プログ

ムであるので，簡単かつ高速に，大量のデータをソートすることができる。又，検索，情報の追加などがこの中で簡単に行なうことができる。本稿で作製する単語レコードファイルは ASCII 標準形式であるので，どのデータベース管理プログラムからでも読み込み可能である。出来上がるファイルはそのデータベース管理プログラム専用のものであり，ASCII ファイルではない。

もう一つの方法は，ASCII ファイルのままソート専用プログラムでソートを行ない，出力も ASCII ファイルで得るものである。著者はこの方法を用いている。[4] 出力ファイルはもとの単語レコードファイルと全く同じ形式で，ただ順番が異なるだけである。このファイルは，再度プログラムで利用することが容易にできるので，辞書様のリスト出力，単語長や音節数の分析を自作のプログラムで行なうことができる。付録4にソート結果，付録5にそれを辞書様にリストしたものを示す。

本稿では，以上のいずれかでソートを行なうこととし，ソートそのものについては省略する。

個々のファイルをソートした後，全てのファイルをマージ（融合[5]）して，一つの大きなファイルを作り，全ての作業を終了する。ソート専用プログラムにはこの様能が付いているのが普通である。データベース管理プログラムを用いる場合は，同じデータベースに単語レコードファイルを次々に読み込めばソートとマージが同時に行なわれるはずである。

著者は，マージ後の最終ファイルも ASCII ファイルで得て，これをプログラムで読み出して，付録5のような形でできるだけスペースをつめて辞書形式で結果をプリントアウトしている。

6. 終わりに

以上で，テキストデータベース作製の基本的部分について述べたが，こうして収集した単語データが，真に利用価値のあるデータベースとなるためには，様々な情報の追加が必要であろう。このような自作のデータベースでは，個人の目的に合わせてそれを容易に行なうことができる。

著者は現在，タガログ語の単語について音節数の情報を加えて，接辞が付加されている長い単語から，語根（多くは2音節からなる）を抽出することを計画中である。

今日様々な文書がデータファイルとしてフロッピーディスクで入手可能になっているが，ここに述べた単語切り出しの方法は，欧文であるならば，あらゆる文書ファイルに応用可能である。普段見なれた文章も，単語にばらばらにしてしまうと全く無機質なものになってしまうが，そこにまた新たな発見があり面白いものである。

本稿では，欧文（ローマンアルファベットで書かれている文書）のみを対象とした。日本文の場合は，単語や文節が必ずしもスペースで区切られていないので，単語の切り出しは全く別な高度な技術を用いなくては不可能である。しかしながら，入力テキストのエディットやマーク付けなどは，テキスト処理の基礎事項として参考になるのではないかと思う。

註

(1) MS-DOS 上のワープロでは，一太郎の .JXW ファイル，P1EXE の .P1 ファイル等が ASCII ファイルである。英文ワープロが無いときはこれらのワープロで入力することは不可能ではない。ASCII ファイルかどうかは，MS-DOS の type コマンドでファイルを画面に出力して確かめることができる。判読可能な文字が出力されればそのファイルは ASCII ファイルである。

(2) これは，ワードスターをはじめ英文ワープロでの文書入力の原則である。ワードスターでは，入力時にキーを打って入れたハイフンをハードハイフン，パラグラフリフォーム時（自動ハイフン on の時）に自動的に入るハイフンと，パラグラフリフォーム時（自動ハイフン off の時）にワープロから促されて行末の単語を分けるために入れたハイフンを，ソフトハイフンという。ソフトハイフンも一度入れればファイル内に常時存在し消えることはなく，画面にも一定のマークで表示されるが，行末に来たときのみプリントアウトされ，その他のときはプリントアウトされない。

ソフトハイフンの内部コードは 9DH で，ハードハイフン（2DH）の MSB を1にした（70H を加えた）ものである。

(3) スペース5個をマークとして使用する方法では，5個以上のスペースがあると，マークとしては[スペース数]－4個分とカウトされてしまうので，注意が必要である。即ち，6個以上のスペースの連続は，いかなる場所にもあってはならない。従って，章のタイトルや Chapter XX のような見出しを，左側にスペースを何個も入れて中央においてはいけない。原文で中央になっていても左に寄せておく必要がある。

(4) 著者は，マイクロプロ社製スーパーソートを使用している。このプログラムはもともと8ビットパソコン上でわずか数十キロバイトのメモリーを用いてソートを行なえるように作られたもので，単機能のプログラムであるが大変汎用製の高いソートプログラムである。今日の16ビット MS-DOS 版も基本的に同じ内容のものであるが，ソートスピードは十分実用になるものである。昨今の何百キロバイトにもなる豪華なプログラムに比べると貧弱に見えるが，何よりも，メモリーが小さくても，どんなシステムでも MS-DOS 上ならば動作可能であることが好ましい。

(5) いくつかの既にソートしてあるファイルを突き合わせ，その順番にしたがって一つの大きなファイルを作ることをマージ（融合）すると言う。マージはソートに比べ簡単で時間もスペースも取らないので，多くのファイルを一度に扱うことができる。ソートとマージを一度に行なうことも不可能ではないが，多くのメモリーと時間を要する。小さなファイルを個々にソートし，後で一度にマージする方が作業能率は良くなる。

付録1　　Turbo Pascal 5.0 (Borland International)　PC9801VM

```
program Divide_Text_inot_Words;

type ASCII = set of #$00..#$7f;

const
   WordChar : ASCII = ['A'..'Z','a'..'z','-',''''];
                        { defines the Word Character Set. }

   DelimiterLn : ASCII=[#$0d,#$0a];          { CR LF  code }
   DelimiterFl = #$2c;          { Comma for field delimiter  }
   MarkPage = '(#)';            { Put Page marker here.      }
   MarkParag= '  ';             { Put Paragraph marker here. }
   MarkComment = '**';          { Put Comment header here.   }

   MarkLeng = 25;    { must be  greater  than the lenghth of
                       any marker.  This defines the size of
                      storage for the markers to be checked.}
var
   infile, outfile :text;
   OneChar         :char;
   MarkStore       :string[MarkLeng];  { keeps Markers.    }

   CountPage,                          { Page counter      }
   CountParag      :integer;           { Paragraph counter }

   WordOn, WordEnd :boolean;           { Word flags        }

procedure InitialVals;
   begin
      WordOn:=false;
      WordEnd:=false;
      MarkStore:='';
   end;

procedure MakeString(OneChar: char);
   begin
      if length(MarkStore) )= MarkLeng then
         repeat
            delete(MarkStore,1,1)
         until length(MarkStore)=MarkLeng-1;
      if OneChar in DelimiterLn
         then MarkStore:=''
         else MarkStore:=MarkStore+OneChar;
   end;

function SearchMarker(Find,Target:string): boolean;
     var i,j,k : integer;
   begin
      SearchMarker:= false;
      i:=pos(Find,Target);
      if i()0 then begin
         j:=length(find);
         k:=length(target);
         if i = k-j+1 then SearchMarker:=true;
         end;
   end;

procedure CheckComment;
   var Comment:string;
   begin
      if     (pos(MarkComment,MarkStore)=1)
         and (length(MarkComment)=length(MarkStore))
         then begin
            readln(infile,Comment);
            writeln(MarkComment,Comment);
            MarkStore:='';
         end;
   end;

procedure SetCounters;
   begin
      write('Count from page? ))) ');readln(CountPage);
      write('  from paragraph? ))) ');readln(CountParag);
      dec(CountParag);
   end;

procedure CheckPage;
   begin
      if SearchMarker(MarkPage,MarkStore)
         then begin
            write('--------------- ');
            writeln('Page: ',CountPage,' - ',CountParag);
            inc(CountPage);
            CountParag:=0;
         end;
   end;

procedure CheckParag;
   begin
      if SearchMarker(MarkParag,MarkStore)
         then inc(CountParag);
   end;
```

```
procedure WriteIndex;
   var s:string[5];
   begin
      str(CountPage,s);
      write  (outfile,DelimiterFl,s);
      str(CountParag,s);
      writeln(outfile,DelimiterFl,s);
   end;

procedure CheckWord;
   begin
      WordEnd:=false;
      if OneChar in WordChar
         then WordOn:=true
         else begin
            if WordOn then  WordEnd:=true;
            WordOn:=false;
         end;
   end;


procedure openinfile(var pinfile:text);
   var FileName, AnyKey :string[16];
       FileExists       :boolean;
   begin
      FileName:='';
      AnyKey:='';
      FileExists :=true;

      repeat
         writeln;
         write('Text File Name ?))) ');readln(filename);
         {$I-}
         assign(pinfile,filename);
         reset(pinfile);
         {$I+}
         FileExists:=(IOResult=0) and (FileName<>'');
         if not FileExists
            then begin
               writeln('',FileName,'',' does not exist.');
               writeln('Type [RET] key to try again.');
               writeln('Any other key to stop the program.');
               readln(AnyKey);
               if Anykey<>'' then halt;
            end;
      until FileExists;
   end;
```

```
procedure openoutfile(var poutfile: text);
   var filename : string[16];
   begin
      writeln;
      writeln('Type the OUTPUT filename ');
      writeln;
      writeln('Example:   CON for CRT');
      writeln('           PRN for Printer');
      writeln('           Other names for the disk files');
      writeln('           [RET] key only for CRT');
      writeln;
      write  ('               ?))) ');readln(filename);

      assign(poutfile,filename);
      rewrite(poutfile);
    end;

{ Main Program }

BEGIN
InitialVals;
OpenInfile (infile);
OpenOutfile(outfile);
SetCounters;

repeat
   read(infile,OneChar);
   MakeString(OneChar);
   CheckComment;
   CheckParag;
   CheckWord;

   if WordOn then write(outfile,OneChar);
   if WordEnd then WriteIndex;

   CheckPage;

until eof(infile);

close(outfile);close(infile);

writeln('-------------------------------- ');
writeln('*** The end of the program.  *** ');
writeln('Press any key.');
readln;

end.
```

付録2

```
**POM 515.7 -- 516.2
**9. The Shot
**Title: A Prayer for Owen Meany
**Author: IRVING, John
**Publisher: Ballantine Books
**Place: New York
**Year: 1989
**Last correction 911121
**1 page mark  5 paragraph marks
```

'Do you actually expect me to wander the world as if I were an addlepated bald woman escaped from the circus?' she would say.

'Missus Wheelwright__where did you put your wigs?' the women would ask her.

'Are you actually accusing me of intentionally desiring to look like the lunatic victim of a nuclear disaster?' my (#) grandmother would ask them. 'I would rather be murdered by a maniac than be bald!'

More wigs were bought; most__but by no means all__of her old wigs were found. When Grandmother especially disliked a wig, she would retire it in the rose garde by submerging it in the birdbath.

And when the Poggios continued to send total...

付録3

```
Do,515,7
you,515,7
actually,515,7
expect,515,7
me,515,7
to,515,7
wander,515,7
the,515,7
world,515,7
as,515,7
if,515,7
I,515,7
were,515,7
an,515,7
addlepated,515,7
bald,515,7
woman,515,7
escaped,515,7
from,515,7
the,515,7
circus,515,7
she,515,7
would,515,7
say,515,7
Missus,515,8
Wheelwright,515,8
where,515,8
did,515,8
you,515,8
put,515,8
ask,516,0
them,516,0
I,516,0
would,516,0
rather,516,0
be,516,0
murdered,516,0
by,516,0
a,516,0
maniac,516,0
than,516,0
be,516,0
bald,516,0
More,516,1
wigs,516,1
were,516,1
bought,516,1
most,516,1
but,516,1
by,516,1
no,516,1
means,516,1
all,516,1
of,516,1
her,516,1
old,516,1
wigs,516,1
were,516,1
found,516,1
When,516,1
```

付録4

```
a,515,9
a,516,0
a,516,1
accusing,515,9
actually,515,7
actually,515,9
addlepated,515,7
all,516,1
an,515,7
And,516,2
Are,515,9
as,515,7
ask,515,8
ask,516,0
bald,515,7
bald,516,0
be,516,0
be,516,0
birdbath,516,1
bought,516,1
but,516,1
by,516,0
by,516,1
by,516,1
circus,515,7
continued,516,2
desiring,515,9
did,515,8
disaster,515,9
disliked,516,1
most,516,1
murdered,516,0
my,515,9
no,516,1
nuclear,515,9
of,515,9
of,515,9
of,516,1
old,516,1
Poggios,516,2
put,515,8
rather,516,0
retire,516,1
rose,516,1
say,515,7
send,516,2
she,515,7
she,516,1
submerging,516,1
than,516,0
the,515,7
the,515,7
the,515,8
the,515,9
the,516,1
the,516,1
the,516,2
them,516,0
to,515,7
to,515,9
```

your,515,8
wigs,515,8
the,515,8
women,515,8
would,515,8
ask,515,8
her,515,8
Are,515,9
you,515,9
actually,515,9
accusing,515,9
me,515,9
of,515,9
intentionally,515,9
desiring,515,9
to,515,9
look,515,9
like,515,9
the,515,9
lunatic,515,9
victim,515,9
of,515,9
a,515,9
nuclear,515,9
disaster,515,9
my,515,9
grandmother,516,0
would,516,0

Grandmother,516,1
especially,516,1
disliked,516,1
a,516,1
wig,516,1
she,516,1
would,516,1
retire,516,1
it,516,1
in,516,1
the,516,1
rose,516,1
garde,516,1
by,516,1
submerging,516,1
it,516,1
in,516,1
the,516,1
birdbath,516,1
And,516,2
when,516,2
the,516,2
Poggios,516,2
continued,516,2
to,516,2
send,516,2
total,516,2

Do,515,7
escaped,515,7
especially,516,1
expect,515,7
found,516,1
from,515,7
garde,516,1
grandmother,516,0
Grandmother,516,1
her,515,8
her,516,1
I,515,7
I,516,0
if,515,7
in,516,1
in,516,1
intentionally,515,9
it,516,1
it,516,1
like,515,9
look,515,9
lunatic,515,9
maniac,516,0
me,515,7
me,515,9
means,516,1
Missus,515,8
More,516,1

to,516,2
total,516,2
victim,515,9
wander,515,7
were,515,7
were,516,1
were,516,1
Wheelwright,515,8
When,516,1
when,516,2
where,515,8
wig,516,1
wigs,515,8
wigs,516,1
wigs,516,1
woman,515,7
women,515,8
world,515,7
would,515,7
would,515,8
would,516,0
would,516,0
would,516,1
you,515,7
you,515,8
you,515,9
your,515,8

付録5

a 515-9;516-0,1 [3]
accusing 515-9
actually 515-7,9 [2]
addlepated 515-7
all 516-1
an 515-7
And 516-2
Are 515-9
as 515-7
ask 515-8;516-0 [2]
bald 515-7;516-0 [2]
be 516-0s[2]
birdbath 516-1
bought 516-1
but 516-1
by 516-0,1s[3]
circus 515-7
continued 516-2
desiring 515-9
did 515-8
disaster 515-9
disliked 516-1
Do 515-7
escaped 515-7
especially 516-1
expect 515-7
found 516-1
from 515-7
garde 516-1
grandmother 516-0,1[2]
her 515-8;516-1 [2]
I 515-7;516-0[2]
if 515-7
in 516-1s[2]
intentionally 515-9
it 516-1s[2]
like 515-9
look 515-9
lunatic 515-9
maniac 516-0
me 515-7,9[2]
means 516-1
Missus 515-8
More 516-1
most 516-1
murdered 516-0
my 515-9
no 516-1
nuclear 515-9
of 515-9s;516-1 [3]
old 516-1
Poggios 516-2
put 515-8
rather 516-0
retire 516-1
rose 516-1
say 515-7
send 516-2
she 515-7;516-1 [2]
submerging 516-1
than 516-0
the 515-7s,8,9; 516-1s,2[7]
them 516-0
to 515-7,9;516-2 [3]
total 516-2
victim 515-9
wander 515-7
were 515-7;516-1 s[3]
Wheelwright 515-8
When 516-1,2[2]
where 515-8
wig 516-1
wigs 515-8;516-1 s[3]
woman 515-7
women 515-8
world 515-7
would 515-7,8; 516-0s,1[5]
you 515-7,8,9[3]
your 515-8